Prefix-Based Popularity Prediction
In Named Data Networking

Bing Han, Xiaofei Wang, Ted “Taekyoung” Kwon, Yanghee Choi
Seoul National University, Korea

Dung Ong Mau, Min Chen
School of Computing Science and Technology,
Huazhong University of Science and Technology, China

AsiaFi 2013 Summer School  August 10, 2013
Outline

- Introduction
- Design of algorithm
  - Prefix tree
  - Lifetime
- Process in cache
- Trace-driven Evaluation
- Conclusion and Future work
Introduction (1)

- To suffer network congestion and large delay in current Internet
  - Named Data Networking (NDN)
- To focus on cache improving in NDN, but ignore effectivity of content in cache
  - Least Recently Used (LRU) & Least Frequently Used (LFU)
- One case:
  - <Gangnam Style> from PSY, over 1.7 billion hits in YouTube
  - <Gentleman> from PSY, over 0.5 billion hits in YouTube

The popular content publishers always make popular content?
Introduction (2)

- How to improve the popular content’s effective?
  - To be stored in the cache as long as possible
  - To be predicted based on cache history

Prefixed-based Popularity Prediction (PPP) Algorithm
Prefixed-based Popularity Prediction Algorithm

- To cache popular content for a global optimization of the cache distribution and predict based on previous prefix statistics
  - Prefix Tree (PT)
  - Lifetime
    - Based on the predicted popularity
  - Based on Power law distribution

An example power-law graph
Prefix Tree in cache

- To determine popularity levels and give suitable lifetime for each content
- Include of prefix’s name and prefix’s counter

Threshold
- A value which is used to screen out the unpopular content
Lifetime in PT

- To determine which content is popular in order to choose to cache the popular content in the cache.
- If the cache space is full, content which short lifetime will be deleted.
- The life time $t_l$ of each content is calculated by:

$$t_l = t_u + \sum_{i=1}^{n} (w[i] \times C_p[i])$$

- For instance:

Party3’s $t_l = 60 + 10 \times 0.3 + 10 \times 0.5 + 10 \times 0.7 + 7 \times 1.0 = 22$
Illustration of PT’s working process

Interest search Prefix Tree

Completely match

yes

no

Check CS

Exists in CS?

yes

no

Forward to neighbors

When the content arrives

Deliver from CS

Update/insert new content

Deliver the content from Cache

Update Cache

Update PT
Evaluation Settings (1)

- **Trace-driven evaluation:**
  - Trace is collected for 77 days from Feb. 14 to May 1, 2011
  - [http://mmlab.snu.ac.kr/traces/bundling/](http://mmlab.snu.ac.kr/traces/bundling/)
  - Trace analysis is not completely finished because of time limit
  - 2 kinds of replacement policy: LRU and PPP
  - 100000 contents, total size is 120TB, 8 million requests,
  - Cache size: from 1 TB to 10 TB
  - Threshold: from 1000 to 10000
Simulation:

- OPNET Modeler 16.0 with NDN
- 3 kinds of replacement policy: LRU at node 1, LFU at node 2, PPP at node 3
- Each node is connected with 25 end-user clients in one side, 3 NDN nodes are connected to the file server which stores 50000 files in the other side
- File size: 1Mb, catalog size: $5 \times 10^5$ Mb,
- Cache size: 300Mb, 500Mb, 800Mb, 1000Mb, relative cache sizes: 0.06%, 0.1%, 0.16%, 0.2%, uses send IntPk in each 5s
Trace Analysis (1)

PPP outperform with higher hitting rate

The different hitting rate value reduced when cache size increase
Trace Analysis (2)

The state of PT size with varied threshold

PT size is reduced across the threshold increase

PT has little overhead (future work: cooperative caching)
Trace Analysis (3)

The multiple threshold policies of PT with varied Cache sizes

- Shorter threshold gain higher hitting rate with larger CS

The multiple Cache size policies of PT with varied PT threshold

- Larger cache size gain higher hitting rate
Trace Analysis (4)

PPP is more effective for cache utilization when CS is smaller.

PPP is no effective for cache utilization when CS is too larger.
Simulation (1)

Higher hitting rate can be gotten when we expand a cache volume

PPP get the highest hitting ratio with the simulation time increase
Simulation (2)

PPP always outperform with higher hitting rate in all situations

PPP gain much better performance and hitting rate is largely concentrated in the most popular contents during the skewness factor increase.
Conclusion and Future work

- New type of cache decision and replacement policy applied for NDN
- PPP algorithm can achieve higher hitting rate and effective caching
- PPP algorithm can efficiently improve the caching performance higher than LRU(LFU)

- Share the PT to neighbors to reduce the overhead
  - Cooperative Caching
- Try to improve the PPP algorithm into NDN